

Fusion

Research



Fusion Research, Inc.

Two-way IP-based Protocol

For

Fusion Media Servers
Updated February 2011

Overview of Fusion Interface Protocol:

The Fusion video and audio servers have unique protocols for video and audio so when writing an interface you must review and understand each set of instructions as they are different from each other. Although the user experience is seamless and fluid when switching between audio and video on the system, the products were developed separately over a several year period so the protocols and setup are different.

This document covers the video interface first in the first section before covering the audio server interface in the later section. There are several additional resources for developing your interface including an excel spreadsheet, which includes a complete set of additional commands for the video interface including a set of IR commands.

In addition to protocol and command documents, there are two test programs that can be helpful to understand and verify your interface; one for both the video and audio portions of the Fusion Media Server. All of the additional documents and test programs can be downloaded from Fusion's website www.fusionrd.com, under the "download" sections of the site.

Finally, Fusion encourages you to call in for any technical support issues you may have while writing your interface to our products. We have developed all our technologies in-house and have staff available to help explain or guide you through the process. You may contact us through our main number at **(925) 217-1233 Monday through Friday 8AM to 5PM Pacific Time.**

Fusion Video Server Simple Control API

For Crestron, AMX and other 3rd party control systems. (Audio Protocol in section 2 of this document).

This document explains how to communicate with Fusion Research's Fusion Video Server products. This is a simple control API intended for 3rd party use to create user interfaces to control and manage a video server.

Definitions

Server	Fusion Video Server product
Client	The 3 rd party control system
Request	A command or query sent from the client to the server
Response	A message sent from the server to a client in response to a request
Notification	A message sent from the server to a client spontaneously
GUID	A unique ID for items in the system. This is a 40-character string.

Connectivity

All communication is handled through TCP. The server will listen and accept connections on TCP port 4441. The Client may open as many TCP connections as it wishes. The server will treat each independently and identically. The server will not close these connections under ordinary circumstances, but the client must be prepared for an unexpected connection termination, and willing to re-establish the connection as necessary.

Summary

The server will...

- Accept TCP connections on TCP port 4441
- Accept any number of connections
- Treat all connections identically
- Not close any connection

The client must...

- Be able to handle unanticipated connection loss

Protocol

All communication is simple ASCII text. A client may send commands or queries to the server. All client requests will receive a response. The server may also send spontaneous status messages or notifications to inform the client about changing conditions on the server. Notifications are optional and off by default. Notification status (On/Off) is specific to a connection and must be set for each connection when established. Once turned on, notifications may be sent at any time, even between a request and a response. All messages (Request, Response and Notification) are formatted with a header, body and terminator.

Message Header

A message header contains an unknown number of characters that represent the following: A command, a PlayerHWSN and a separator (to separate the header from the body of the message).

There are two command characters for a request:

- ! Indicating a command (i.e. start playing this movie)
- ? Indicating a query (i.e. list videos in system)

There is a single command character for a response:

~ Indicating response to a command (i.e. Stopped)

There is a single command character for a notification:

* Indicating notification or status update (i.e. Player state)

The PlayerHWSN (video player hardware serial number) is a string indicating the player we want to refer to. A PlayerHWSN is always required, but does not have to be valid for database queries.

Finally a colon (:) separates the header from the body.

Examples

!{20-6}: Send command for player {20-6}

~{20-6}: Result of command to player {20-6}

?0020321: Query player 0020321

*0020321: Notification from player 0020321

Message Terminator

All messages will be terminated with the ASCII Carriage Return (decimal 13).

Message Bodies

Message bodies are specific to commands, responses and notifications.

Response Messages

In general, responses will indicate success or failure with simple text. OK indicates the command completed successfully while Error will indicate a failure and may be followed by a space and explanation.

Examples:

~021:OK<CR>

The command was completed successfully.

~65-A:Error<CR>

The command could not be performed.

~2144:Error The zone is not available<CR>

Command not performed (and here's why).

~X:OK <more data> <CR>

Response to a query.

On occasion, a successful response will also contain additional information, in response to a query, for example. In this case, the OK will be followed by a space and additional data.

Notification Messages

The server sends notifications when things change on the server or on a player (if Notify is On). All notification bodies are formatted as a Key=Value pair of data. For example, when a Player starts to play, a notification will come in as "Transport=Running" to indicate the zone's transport has changed to the Play state (as opposed to the Pause or Stop state). A client should be prepared to ignore notifications which it does not understand (such as unusual keys).

This is a list of possible Keys and Values for each key.

Key	Value	Description
Notify	On	Notifications from the server are On. i.e. *02:Notify=On<CR>
	Off	Notifications from the server are Off.
ProgramTitle	Title of a song	This is the title of the playing song. i.e. *01:ProgramTitle=Star Wars IV<CR>
ProgramGUID	A GUID	This is the GUID of the playing video. The GUID can be used to retrieve cover art for the video. An http request can be made of the server on port 4440 using the following URL: /akemi/video/art.cgi?GUID=<GUID>&Height=<height>&Width=<width> where the height and width parameters are optional.
Transport	Running	Player is now playing i.e. *05:Transport=Running<CR>
	Paused	Player is now paused
	Stopped	Player is now stopped
Duration	Time in minutes	This is the duration of the video in minutes. i.e. *01:Duration=121 <CR>
Position	Time in minutes	This is the position in the video in minutes. i.e. *01:Position=43 <CR>

Adjusting projectors and screens for movie formats

A note about screen formatting: Often an integrator will want to know the screen format of a movie when it starts to perform automation tasks such as adjusting screen masking or manipulating projector lenses. This can be easily accomplished by using a two-way interface. To do so, do the following:

- 1) Establish a control connection to the server and turn notification on to receive events from the system.
- 2) Watch for the ProgramGUID event, which will be received when a video is selected for playback by the customer.
- 3) Extract the Program GUID from the event, and request Info on the program.
- 4) Read the returned information and extract the Aspect Ratio from the response.

Command Messages

All commands should be sent through the previously established command system. See the external spreadsheet for more information.

If this is not possible, commands can be sent to this interface that will "bridge" them for you to the already established interface. To do this, you use the Key of "Perform". The value would be a string comprised of Key=Value pairs separated by ampersands (&). Please refer to the external spreadsheet for more information. (Note: Requires Akemi Video Server version 8.9.6.0 or newer)

Key	Value	Description
Perform	URL parameter list	Forward the URL parameters in the proper format for player control.

Examples

!0010002:Perform=Group=Navigation&Prop=NavAction&Op=Set&Value=Down
 This will tell player 0010002 to perform a cursor down.

!0010002:Perform=Group=Navigation&Prop=NavAction&Op=Set&Value=Up
 This will tell player 0010002 to perform a cursor up.

Fetching Cover Art

Cover art for a video program can be retrieved via HTTP request by asking the server on port 4440 for art given a GUID. The URL would be /akemi/video/art.cgi?GUID=[GUID] where [GUID] is the GUID of the video of interest. The server supports additional parameters of Height and Width for scaling of the image. If both are omitted, the original image is returned in original size. If either is provided, the image is resized, but the aspect ratio is maintained. If both are provided, the image is stretched to fit.

Example: http://127.0.0.1:4440/akemi/video/art.cgi?GUID={67E24B28-F4A1-4157-AEFF-0B8EB2B26529}

Query Messages

Queries are used to navigate the video data. The data is presented as a single list of video programs. You start by asking for a list of programs. This list should be limited. You can then page through the list to browse the data. Again, this request is in the form of a Key=Value pair.

This is a list of possible Keys and Values for each key.

Key	Value	Description
Transport		Returns the current transport state (Running, Paused or Stopped) i.e. ?02:Transport<CR>
List	Programs	List the programs. The response is specially formatted and will be discussed later.
List(x,y)	Programs	This does the same thing as the standard List command except the x and y are integer numbers which indicate the first entry to return and the total number of entries to return. Entries are zero (0) indexed, so asking for List(0/8) will return the first 8 entries, where List(8/8) will return the next 8 entries. i.e. List(0/9) Programs
ListA(x,y)	Programs	See List(x,y) except here X is a letter.
Info	GUID	This returns information for a program specified by GUID. The format of the response will be discussed later.

Response format for List commands

After the OK (assuming success), there will be a space, followed by the following:

```
Programs<TAB>First record number<TAB>Total record count<LF>
GUID of video<TAB>Title of Video <LF>
GUID of video<TAB>Title of Video <LF>
GUID of video<TAB>Title of Video <LF>
```

Here, lines are separated by LF (decimal 10) and fields in a line are separated by tabs (decimal 9). The first line contains the following fields: Programs (indicating we are receiving a list of programs), The record number (zero based) of the first entry returned and finally the total number of entries available.

Each line thereafter represents an entry with the following fields: GUID of the video, Name of the video (title).

Response format for Info command

After the OK (assuming success), there will be a space, followed by the following:

```
GUID of Program<TAB>Title of program<TAB>ReleaseDate<TAB>
RunningTime<TAB>AspectRatio<TAB>Rating<TAB>Genres<TAB>
Actors<TAB>Directors<TAB>PlotSummary<TAB>Plot<TAB>Review<CR>
```

Genres, Actors and Directors are semicolon separated lists.

Sample Translation shown on next page

Sample Transaction (Video)

Note, <CR> represents a Carriage Return (decimal 13), <LF> represents a Line Feed (decimal 10) and <TAB> represents a Tab character (decimal 9).

<p>Client ?{TEST_2}:List(0,10)=Programs</p>	<p>Server ~{TEST_2}:OK Programs 0 9 {67E24B28-F4A1-4157-AEFF-0B8EB2B26529} Discoverers_1080 {B91FD251-9EB9-4FCD-B048-219EE9702086} Identity {677F446A-52C7-40D5-AA70-5E77F405C5F9} Underworld: Evolution {A5F56F41-AF43-4670-A181-69053AFB36BC} Hellboy {42D46EED-8FE5-46AA-9F30-525D20695EDF} Resident Evil: Apocalypse {B1873911-9083-4B2E-8E37-5F6AB1D7C759} Blue Alien Demo {0668D193-06A3-49D9-BC9F-A4D649C2581B} Flak_DivX_50 {FB7495F2-009B-4498-B187-8FB1CDCDCE05} Teenage Mutant Ninja Turtles 480p {GIRLFIGHT} Girlfight</p>
<p>?{TEST_2}:Info={B91F... 86}</p>	<p>~{TEST_2}:OK {B91FD251-9EB9-4FCD-B048-219EE9702086} Identity 2002-06-14 116 2.35:1 R Horror;Horror;Horror Mangold, James;John C. McGinley;Pruitt Taylor Vince;Marshall Bell;Carmen Argenziano;Rebecca De Mornay Liman, Doug Stranded at a desolate Nevada motel during a nasty rainstorm, ten strangers become acquainted with each other when they realize that they're being killed off one by one. Strangers from all different walks of life: a limo driver escorting a movie star, parents with a young son, a cop transporting a convict, a prostitute, a young couple, and a motel manager are caught up in a nasty rainstorm, stuck at a motel in desolate Nevada. Soon they realize they may be at the motel for another reason when one by one, people start getting killed off. As tensions flare and fingers are pointed, they have to get to the bottom of why they're there. Meanwhile in an undisclosed location, a psychiatrist is trying to prove the innocence of a man accused of murder in an eleventh hour trial. How these two through-lines are related can only be found in Identity. Freely adapted from Robert Ludlum's 1980 bestseller, The Bourne Identity starts fast and never slows down. The twisting plot revs up in Zurich, where amnesiac CIA assassin Jason Bourne (Matt Damon), with no memory of his name, profession, or recent activities, recruits a penniless German traveler (Run Lola Run's Franka Potente) to assist in solving the puzzle of his missing identity. While his CIA superior (Chris Cooper) dispatches assassins to kill Bourne and thus cover up his failed mission, Bourne exercises his lethal training to leave a trail of bodies from Switzerland to Paris. Director Doug Liman (Go) infuses Ludlum's intricate plotting with a maverick's eye for character detail, matching breathtaking action with the humorous, thrill-seeking chemistry of Damon and Potente. Previously made as a 1988 TV movie starring Richard Chamberlain, The Bourne Identity benefits from the sharp talent of rising stars, offering intelligent, crowd-pleasing excitement from start to finish. --Jeff Shannon</p>
<p>!{TEST_2}:Notify=On</p>	<p>~{TEST_2}:OK</p>

Fusion Audio Server Simple Control API

For Crestron, AMX and other 3rd party control systems.

This document explains how to communicate with Fusion Research's Fusion Audio Server products. This is a simple control API intended for 3rd party use to create user interfaces to control and manage an audio server.

Definitions

Server	Fusion Audio Server product
Client	The 3 rd party control system
Request	A command or query sent from the client to the server
Response	A message sent from the server to a client in response to a request
Notification	A message sent from the server to a client spontaneously
GUID	A unique ID for items in the system. This is a 40 character string.

Connectivity

All communication is handled through TCP. The server will listen and accept connections on TCP port 4724. The Client may open as many TCP connections as it wishes. The server will treat each independently and identically. The server will not close these connections under ordinary circumstances, but the client must be prepared for an unexpected connection termination, and willing to re-establish the connection as necessary.

Summary

The server will...

- Accept TCP connections on TCP port 4724
- Accept any number of connections
- Treat all connections identically
- Not close any connection

The client must...

- Be able to handle unanticipated connection loss

Protocol

All communication is simple ASCII text. A client may send commands or queries to the server. All client requests will receive a response. The server may also send spontaneous status messages or notifications to inform the client about changing conditions on the server. Notifications are optional and off by default. Notification status (On/Off) is specific to a connection and must be set for each connection when established. Once turned on, notifications may be sent at any time, even between a request and a response. All messages (Request, Response and Notification) are formatted with a header, body and terminator.

Message Header

A message header contains four (4) characters that represent the following: A command, a zone (or output) number or player serial number and a separator (to separate the header from the body of the message).

There are two command characters for a request:

- ! Indicating a command (i.e. start playing this song)
- ? Indicating a query (i.e. list tracks in an album)

There is a single command character for a response:

~ Indicating response to a command (i.e. Stopped)

There is a single command character for a notification:

* Indicating notification or status update (i.e. Position in track)

The zone number is a two-digit number (zero padded) indicating the audio output (or zone) of the server. At present, Fusion audio servers ship with one or five zones, but this may change in the future. Depending on the server, valid zone identifiers are 01, 02, 03, 04 or 05. 00 is a special zone number referring to the server itself.

Finally a colon (:) separates the header from the body.

Examples

!01: Send command for zone 1

~01: Result of command on zone 1

?02: Query zone 2

*04: Notification from zone 4

!0020350: Send command for "zone zero" of the player with serial number 0020350.

Message Terminator

All messages will be terminated with the ASCII Carriage Return (decimal 13).

Message Bodies

Message bodies are specific to commands, responses and notifications.

Response Messages

In general, responses will indicate success or failure with simple text. OK indicates the command completed successfully while Error will indicate a failure and may be followed by a space and explanation.

Examples:

~01:OK<CR>

The command was completed successfully.

~05:Error<CR>

The command could not be performed.

~04:Error The zone is not available<CR>

Command not performed (and here's why).

~03:OK <more data> <CR>

Response to a query.

On occasion, a successful response will also contain additional information, in response to a query, for example. In this case, the OK will be followed by a space and additional data.

Notification Messages

The server sends notifications when things change on the server or in a zone (if Notify is On). All notification bodies are formatted as a Key=Value pair of data. For example, when a Zone starts to play, a notification will come in as "Transport=Play" to indicate the zone's transport has changed to the Play state (as opposed to the Pause or Stop state). A client should be prepared to ignore notifications which it does not understand (such as unusual keys).

This is a list of possible Keys and Values for each key.

Key	Value	Description
Notify	On	Notifications from the server are On. i.e. *02:Notify=On<CR>
	Off	Notifications from the server are Off.
Title	Title of a song	This is the title of the playing song. i.e. *01:Title=Owner of a Lonely Heart<CR>
Title_Next	Title of a song	This is the title of the song that will play next. i.e. *01:Title_Next=Midnight Rambler<CR>
Artist	Name of an artist	This is the artist of the playing song. i.e. *02:Artist=AC/DC<CR>
Album	Name of an album	This is the album the playing song is from. i.e. *04:Album=Powerslave<CR>
GUID	A GUID	This is the GUID of the playing song. The GUID can be used to retrieve cover art for the song. An http request can be made of the server on port 4720 using the following URL: /akemi/media/art/<GUID>?Height=<height>&Width=<width> where the height and width parameters are optional.
Transport	Play	Zone is now playing i.e. *05:Transport=Play<CR>
	Pause	Zone is now paused
	Stop	Zone is now stopped
Repeat	On	Repeat mode is on for the zone. i.e. *03:Repeat=On<CR>
	Off	Repeat mode is off for the zone.
Random	On	Random mode is on for the zone. i.e. *02:Random=Off<CR>
	Off	Random mode is off for the zone.
Append	On	Append mode is on for the zone. Append mode allows additional music to be added to the list of playing songs without replacing the list. When Append is off, each command to select music will replace what is playing. With it on, the commands will add to the music to be played.
	Off	Append mode is off for the zone. i.e. *01:Append=On<CR>
Length	Value in seconds	This is the playing time of the song in seconds. If this is not known, the value will be zero (0). i.e. *03:Length=315<CR>
Position	Value in seconds	This is the position in the song we are currently at. i.e. *03:Position=23<CR>

Command Messages

Commands can be sent to the server to control zone playback. Command bodies also take the Key=Value format.

This is a list of possible Keys and Values for each key.

Key	Value	Description
Notify	On	Turn notifications from the server On. i.e. !02:Notify=On<CR>
	Off	Turn notifications from the server Off. Note the Zone is necessary here, but ignored because notification is a per-connection property.
Power	On	Unnecessary... the zone is always "on". It just may not be playing music or have anything to play. i.e. !01:Power=On<CR>
	Off	This stops playback and clears the zone's list of music to play. It does not actually render the zone incapable of operation.
Transport	Play	Play the current song, or resume playback.
	Pause	Pause playback (or resume)
	Stop	Stop playback.
	Prev	Play the last song or return to the start of the current song. This is transitory, and the transport will return to it's previous state after completion. This should only be used while playing.
Random	Next	Play the next song. This is transitory, and the transport will return to it's previous state after completion. This should only be used while playing. i.e. !04:Transport=Pause<CR>
	On	Turn random on for the zone
Repeat	Off	Turn random off for the zone
	On	Turn repeat on for the zone
Append	Off	Turn repeat off for the zone
	On	Turn append on for the zone
Play	Off	Turn append off for the zone i.e. !02:Append=On<CR>
	GUID	Play this item. This can be a folder or track. Note some folders cannot be played as they contain too many tracks (i.e. the Genres folder)

Query Messages

Queries are used to navigate the music data. The data is presented in a folder structure. Folders contain other folders and files. You start by asking for a list of folders and letting the user select one, then asking for it's contents until the user selects a folder or file to play. Again, this request is in the form of a Key=Value pair.

This is a list of possible Keys and Values for each key.

Key	Value	Description
Transport		Returns the current transport state (Play, Pause or Stop) i.e. ?02:Transport<CR>
Random		Returns the current "Random" state (On/Off)
Repeat		Returns the current "Repeat" state (On/Off)
Append		Returns the current "Append" state (On/Off)
List	GUID	List the contents of the Folder specified by the GUID. The response is specially formatted and will be discussed later. The GUID is a GUID of a folder. There are certain pre-defined GUIDs that can always be used.
List(x,y)	GUID	This does the same thing as the standard List command except the x and y are integer numbers which indicate the first entry to return and the total number of entries to return. Entries are zero (0) indexed, so asking for List(0/8) will return the first 8 entries in the folder, where List(8/8) will return the next 8 entries. i.e. List(0/9) Genres
ListA(x,y)	GUID	See List(x/y) but here the X is a letter.

Note: For all List commands, the GUID can be replaced with one of the following words: Album, Artist, Genre, Playlist. These are "pre-defined" to always exist in the system.

Pre-defined GUIDs	Alternate Name	Meaning
{FOLDER-ROOT-MUSIC-ALBUM}	Albums	List all Albums
{FOLDER-ROOT-MUSIC-ARTIST}	Artists	List all Artists
{FOLDER-ROOT-MUSIC-GENRE}	Genres	List all Genres
{FOLDER-ROOT-MUSIC-PLAYLIST}	Playlists	List all Playlists

Response format for List commands

After the OK (assuming success), there will be a space, followed by the following:

```
GUID of folder<TAB>Name of Folder<TAB>GUID of folder parent<TAB>First record
number<TAB>Total record count<LF>
GUID of entry<TAB>Name of Entry <TAB>Kind of entry<LF>
GUID of entry<TAB>Name of Entry <TAB>Kind of entry<LF>
GUID of entry<TAB>Name of Entry <TAB>Kind of entry<LF>
```

Here, lines are separated by LF (decimal 10) and fields in a line are separated by tabs (decimal 9). The first line contains the following fields: GUID (of the folder), Name (of the folder), GUID of the folder's parent (if applicable), The record number (zero based) of the first entry returned for the folder) and finally the total number of entries in the folder.

Each line thereafter represents an entry in the folder with the following fields: GUID of the entry, Name or the entry, Kind of the entry. Kinds are folder, audio/mp3 and audio/wav.

Control of Players (Zone Zero)

This API is designed to control audio output from an audio server (a one or five output music or video server, or a one output Genesis video server/player). In addition, all video players support what is known as “Zone Zero”. This allows the player’s audio to be controlled through this API with a few caveats.

- 1) This API will NOT control the player’s user interface.
- 2) The player’s behavior (random, repeat and/or append modes) as managed through this API is NOT the same as it’s behavior through it’s user interface. In other words, putting the player into “append mode” through this API will NOT allow the user to “append” tracks from the user interface. If the user wants to do that, they must do that through the user interface.
- 3) The player only has one playlist, and manipulating it through this API or the user interface will result in the same outcome, however, the user interface may not reflect changes made through this API immediately.

To control a player’s “Zone Zero”, simply use the player’s serial number as the zone number in all commands. Note: The serial number is a string, not a number, and so 0020350 is NOT the same as 20350. Enter the serial number exactly as it appears on the device’s serial number tag.

